UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/803,514	03/08/2001	Sridhar Obilisetty	026507-000300US	6110

20350 7590 04/30/2009

TOWNSEND AND TOWNSEND AND CREW, LLP
TWO EMBARCADERO CENTER
EIGHTH FLOOR
SAN FRANCISCO, CA 94111-3834

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
04/30/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	09/803,514	OBILISETTY, SRIDHAR	
	Examiner	Art Unit	
	TUAN A. VU	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *2/20/09*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-6 and 8-25* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-6,8-25* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date. _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 2/20/09.

As indicated in Applicant's response, claims 1, 13, 25 have been amended. Claims 1-6, 8-25, 37-38 are pending in the office action.

Claim Rejections - 35 USC § 112

2. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

3. Claim 25 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Claim 25 recites 'each of said text files defines a component of said application' and there is no sufficient antecedent basis for 'said application' and this will be treated as any application.

4. Claim 19 recites 'wherein said plurality of text files … are particular to said computer system'. The degree of how particular one object is to another or to subject would largely dependent upon how the quality of being particular is measured or weighted, and this is insufficiently provided from the claim; and the 'are particular to' is deemed a relative concept, not a definite and objective degree based on which one can learn on the full measure of this 'particular to' quality/characteristic of the "text files" vis-à-vis a rather broad concept like a 'computer system'; hence the above language is indefinite for failing to clearly establish the *metes and bounds* of this 'particular to' for its indefiniteness would not enable one to make use or implement this invention without undue experimentation.

5. Claims 1-6, 8-25, 37-38 are rejected for lack of antecedent basis in the language recited as 'independent from *said program and* independent *from the program interacting with the server*' (step b of claims 1, 13, 25). Since the only program interacting with the server is *said program* or *resident program*; the entity phrased as "the program interacting with the server" has no expected basis as to justify why it is there in the context of the claim; and this limitation will not be given merits. The dependent claims are also rejected for depending on the above base claims.

6. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

7. Claims 1-6, 8-25, 37-38 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention.

Specifically, claims 1, 13, and 25 recite '*compiling automatically* … a combination of said *updated versions of said text files to create said application* wherein said application is *stored* on said client computer *in a compiled form* so as to be executable … in a runtime environment …' (cl. 1, 5th para; cl. 13, 6th para; cl. 25, 6th para); and it is observed that this automatically compiling of (combination of) updated text files to create a compiled and executable form of the application (stored on the client computer) is not provided with proper description or reasonably enabled by the Specifications. The Disclosure mentions that no editing or recompiling is needed (Specifications, pg. 18 bottom half) when downloaded XML files

enable user application(s) to be updated or seamlessly updated (Specifications, middle pg. 20) as this would reduce costs associated with application upgrades. The files being gathered by agent 205 are text files used to define GUI, business logic, communication preferences, data management, exchange logic, business rules or workflow, content of data or structural relationships thereof (Specs: middle pg. 17) and the text files are modifiable within a user interface (bottom pg 20 to top pg. 21) and application like Work Processing can be defined by XML business logic files (bottom pg. 17) that can be uploaded to a server. Clearly, the so-called application as disclosed amounts to application in a form that does not require re-compilation even after the business logic/rules or XML metadata have been incorporated for seamlessly updating the application (e.g. Word processing); and agent 205 by assembling the XML files to automatically make the application up-to-date cannot be construed as a compiler that translates updated versions of XML files into a result clearly perceived as a newly created compiled form of the application that is created/stored executable. Word processing is disclosed as being automatically updated with information inside of XML text files; and integrating update metadata (or data defining a business logic, GUI preferences, or communication rules) into application like Word Processing, even this is asynchronous from current usage of Word Processing by a user, cannot be same as a compiler compiling text files to CREATE a compiled form that would be stored as executable file. Without undue experimentation, one of ordinary skill in the art would not know how to create a compiled form of an application (that is never subject to recompiling) such that this form is stored as an executable because absent specificity regarding how text files (or combination of updated versions thereof) are actually compiled, the fact of merely incorporating XML type or meta-definition text into an application by agent 205

does not follow that a compiled form is generated, even more so, when the invention underlines that no re-editing or re-compiling is needed. The agent is not described as a programming language compiler, nor is there any step of code translating anywhere in the actions taken by this agent, whose role is limited to downloading and upgrading preferences, rule or meta-information of an application. As disclosed, the assembling of definition data based on the downloaded XML files amounts to upgrading (definition data) of an application without any form of recompilation; and upgrade with definition data (without much implementation details regarding how this is done) cannot be construed as compilation to build a new application (i.e. *to create said application*). The application as disclosed has never been *created* from using the downloaded text files when no compiler is evident and when the disclosed agent is to merely assemble definition data (see *updated by downloading* – ABSTRACT of application) for a update purpose; that is, the 'to create' will not be given weight.

To the extent as to reasonably convey what *compiling text files* is all about, the disclosure **is silent** (emphasis added) about how definition data or text file content is translated/extracted by the agent and/or how incorporation (by this very agent) of the extracted definition data is actually changing the internal code structure of the user application (e.g. Modeling/media tool, GUI or MS word); and in this perspective alone, what is being achieved inside this seamless upgrade (see ABSTRACT) by the agent is considered devoid of structure transformation, algorithmic means, credible facts or substantial actions (emphasis added).

Not recompiling an application (or operating without a compiler) as set forth in the Disclosure, the upgrading from above signifies that the client application (e.g. a Word processing application) remains in its original executable form; and the fact of integrating definition data (or

metadata) into a running or non-running application (the application not being recompiled) cannot be equated to compiling so to yield a compiled executable form of one created application, which then can be locally stored for reuse. The inventor is not deemed in possession of a *compiling means or steps* to actually translate a combination of (updated versions of) XML text files **to form an executable** file **which is stored for further execution** (emphasis added). The above 'compiling automatically … combination of updated versions of text files … to create … said application is stored … in a compiled form so as to be executable' is not given full patentable weight; and will be treated as though the application is incorporated with (downloaded) data/information via use of a separate piece of update/integrator code that communicates with a text file provider: any text files derived, compiled, then stored form of reusable executable independent from the original runtime for not being enabled by the Disclosure will not be given patentable merits, with 'create said application' treated as a mere internal upgrade, which is also interpreted in a very broad sense.

Claims 2-6, 8-12, 14-24, 37-38 are also rejected for not curing to the lack of enablement as identified from above.

8. Claims 1, 13, 25 further recite 'executing said application on said runtime environment … such that application remains executable on said runtime … *upon removal of said program from said client computer system*' (cl. 1, 13, 25: last para). Regarding 'said application' being executed, the disclosure is silent above a scenario by which an agent is executing to compile files an executable form of an application (see above), with this form being a result of a compilation of text files, such that the compiled form is from an actual creation and compilation combination. As disclosed, the assembling of definition data based on the downloaded XML files amounts to

upgrading (definition data) of an application without any form of recompilation; and upgrade (via downloading - see Abstract) of definition data (without much implementation specifics) cannot be construed as compilation to build a new application (i.e. *to create said application*). The Disclosure emphasizes that asynchronous use of an application by the user (Specs: pg. 20-21) can be concurrent with agent 205 (a resident program on the client) that checks for updates (for that application) with a communication session with a server, therefore the application always existed and never gets built and created from scratch. More importantly, the asynchronous nature of this application runtime and the communication session convey to one (of ordinary skill in the art) that as session becomes broken, the runtime state of the application currently in use by the user would not be affected (bottom pg. 20). Asynchronous occurrences of processes in lower layers of NW communication/paradigm underlying an upper layer where a application sends out request/messages cannot be equated to a particular event that removes a program from a computer system. The disclosure does not teach that when the agent is uninstalled (or removed) from the client computer (*removal of said program from said client computer system*) the application under the given execution runtime remains an executable in said runtime. Further, the above independency state based on asynchronous execution and communication to retrieve updates does not establish that the running application is independent from (i) *said program* and from (ii) *the program interacting with the server* (see USC 112, second para), since (i) and (ii) appear to be only one program related to one session, transparent from the user runtime, since agent 205 cannot represent both (i) and (ii) -- said program and a program interacting with a server. The disclosure is not clearly provided with a description that

teaches about the event of **actual** removal of agent 205 from the client system **in terms such that:**

A) *the created* and *stored executable* form of the application *compiled by way of* agent 205 -- using downloaded *updated versions* of text files as recited in *step a)* -- remains executable; or

B) that the original application executing independently from the asynchronous session (pg. 20) with a server remains active.

The 'removal of said program from said client computer' will not be given patentable weight, notwithstanding the lack of enabling support regarding **compiling** automatically ... to **create** said application ... **stored** ... in a **compiled form** ... as to be executable' as set forth from above, whereby no weight is given to **creating a** application, to **automatically compiling** into a **stored compiled form** as executable application.

The 'removal' will be treated as though communication with the server is not present or dynamic download is not concurrent.

The dependent claims 2-6, 8-12, 14-24, 37-38 are rejected for not curing to the lack of support for the above language.

Claim Rejections - 35 USC § 103

9. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

10. Claims 1-6, 8-25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Bloch et al, USPN: 2002/0129129 (hereinafter **Bloch**) and further in view of Murray et al, USPN: 6,874,143 (hereinafter Murray).

As per claim 1, Bloch discloses a method for implementing an application on a client computer system, said method comprising steps of:

a) executing a program resident on said client computer system (e.g. para 0050 pg. 5; para 0053-0055, pg. 6; AVM, Install File, starter page - Fig. 3; para 0057-0058, pg. 6 – Note: executing installation code in AVM initializing stage including plugin/starter routines to setup memory or GUI handles – para 0062, pg. 7; Fig. 2-3 – reads on executing program resident on client),

wherein said program comprises instructions for interacting with a server in accordance with receiving at said client computer system a plurality of text files wherein each of said text files defines a component of said application (e.g. *initializes ... downloads* - para 0055, pg. 6; para 0057-58, pg. 6; para 0061, pg. 7; step 70-72, Fig. 5); and

compiling automatically a combination of said updated versions of said text files to create said application (e.g. Fig. 2, 5; para 0044, pg. 4; para 0063 pg. 7; para 0081-0083, pg. 9; *each function belongs to that application* – para 0085 pg. 9 – Note: compiling to create an application treated as incorporating downloaded/upgrade data into an existing application with NO recompilation of the latter – see USC 112 Rejection);

wherein said application is stored on said client computer in a compiled form so as to be executable independent from said program in a runtime environment (para 0084-0085, pg. 9; Fig. 5) independent from said program (e.g. Fig. 5, 8, para 0084-0088; Fig. 9, 10 - Note: *compiled*

form being stored – see USC 112 Rejection – treated as a user application state in which the application being developed after incorporating text files remains executable or deployable by the user environment or platform independently from the earlier download/initializing stage); and

b) executing said application on said runtime environment (e.g. para 0037, pg. 4; para 0044, pg. 5; Fig. 9-10) independent from said program (Note: *said program* given weight only as communication instance by which initial stage of AVM routines download XML) and independent from the program interacting with the server such that said application remains executable (Note: *removal of said program* treated as state in which no download/server-related communication is concurrent with the runtime – see USC 112 Rejection – where the user is validating the intended script elements or modules that make up functions of the application– see para 0097-0101, pg. 11) on said runtime environment upon removal of said program from said client computer system.

Bloch does not discloses executing resident program to further (1) check *automatically and periodically* for updated versions of said text files; (2) receive automatically any updated versions of said text files in response to said program checking for said updated versions when said updated versions are available; then (3) compile *automatically and periodically* using said combination of said updated versions (Note: compiling treated as incorporating to upgrade a current state).

Bloch discloses compiling automatically using a combination of versions of said text files to create said application (e.g. Fig. 2, 5; para 0044, pg. 4) wherein said application is stored on said client computer in a compiled form (para 00100, pg. 11; para 0102, pg. 11; para 0103, pg.

11; para 0105, pg. 12 - Note: executing a script with underlying invoking of procedures reads on compiled form being stored); and further discloses upgrades and fixes and possibility to make use of most recent versions (*upgrades, fixes* - pg. 3, para 0032; *most recent ...version* - pg. 12, para 0107) and observes on the urge for providing latest set of files in accordance to appropriate version of script file or virtual machine files with regard to version (e.g. *receive upgrades as soon as possible* -para 0051-0053, pg. 5-6); hence has taught checking of files and their latest upgrade, as referred to in Bloch as a well-known scheme to hot-synch of updates (each new version) with a particular platform (see para 0010, pg. 1).

Similar to Bloch's approach as to use extensible format to implement definition data in support for a target application on-demand, **Murray**, in a similar framework to develop browser applications with the versions of XML extension type of data, discloses a delivery of extension files (e.g. Fig. 15-16), with developer executing a client-side developing tool in contact with server environment to periodically check for download of latest upgrades for XML-implemented files (*periodically polling* - col. 18 line 40 to col. 19 line 17). In light of the desirability of updating browser/PC application to meet the appropriate virtual machine or execution environments in view of practices such as *hot-synch* mentioned above along with the implicit need to monitor of new version availability in order to maintain compatibility of data needed to process by a browser-based development in light of the resource-restraints imposed upon portable platform (e.g. Bloch: WPDA 111- Fig. 1), it would have been obvious for one of ordinary skill in the art at the time the invention was made to enhance Bloch's endeavor to obtain proper version of files (para 0053, pg. 6) so that using the browser engine utilizes an automatic upgrade-checking mechanism via **periodic** inquiry or polling as taught in Murray in order for the

latest upgraded XML or browser files to be retrieved for compilation (in a corresponding **periodic** scheme) as endeavored in Bloch's use of XML files, whereby extending/developing client or developer's applications in terms of upgraded versions incorporated in the target application as set forth in Bloch. One would be motivated to do so because this periodic remote check/upgrade (as in 1) would enable the executing environment to be provided with the appropriate files (as in 2) according to the version expected for such environment where when upgraded, on such periodic basis, the target application (as in 3) would benefit from the latest set of definition data purported by both Bloch's approach and Murray's browser extension tool from above, while obviating constant use of extraneous storage of files at the target system.

As per claim 2, Bloch discloses XML format (Fig. 1, 2, 4,5).

As per claim 3, Bloch discloses server central source for managing and distributing applications or modifications for applications (e.g. *upgrades, fixes* - pg. 3, para 0032; pg. 5, para 0045-0046; pg. 12, para 0109).

As per claim 4, refer to claim 3 and Bloch's Fig. 1, 2, 4,5.

As per claim 5, Bloch discloses executing an application, sending a request and executing the application in parallel while waiting for response from the request (e.g. … *reports to the Application Handler 302*, … *periodically updates* -- pg. 9, para 0080 – 0082 – Note: resolving a URL with data retrieval while leaving the GUI window on for being updated on tree events changes and notified of download status is equivalent to executing application while waiting for remote response)

As per claim 6, Bloch discloses connectionless application execution (e.g. pg. 8, para 0069; pg. 12, para 0108)

As per claim 8, Bloch discloses modifying application by using a newer text files replacing older files (*upgrades, fixes* - pg. 3, para 0032; *most recent ...version* - pg. 12, para 0107).

As per claim 9, Bloch discloses graphical user interface (e.g. Fig. 6).

As per claim 10, Bloch discloses application being communication preferences for database invocation (e.g. pg. 7, para 0063; Preference Handler 303 - Fig. 4)

As per claim 11, Bloch discloses data management application (e.g. step *52* – Fig. 5; Manager 301 -Fig. 4 - Note: downloading files to assemble manager module reads on application being a management application).

As per claim 12, Bloch discloses component being part of logic of application (pg. 1, para 0012; pg. 4, para 0037).

As per claim 13, Bloch discloses a computer system comprising:

a bus; a computer-readable memory unit coupled to said bus; and a processor coupled to said bus, said processor for executing (Fig. 2) a method for implementing an application comprising:

a) executing a program resident on said computer system (refer to claim 1) wherein said program comprises instructions for interacting with a server in accordance with:

receiving at said client computer system a plurality of text files wherein each of said text files defines a component of said application (refer to claim 1);

compiling automatically a combination of Said updated versions of said text files to create said application (refer to claim 1),

wherein said application is stored on said computer system in a compiled form (refer to claim 1 and USC 112 Rejection) so as to be executable independent from said program in a runtime environment independent from said program; and

b) executing said application on said runtime environment independent from said program and independent from the program interacting with the server (refer to claim 1)such that said application remains executable on said runtime environment upon removal of said program (refer to claim 1 and USC 112 Rejection) from said client computer system.

Bloch does not discloses executing resident program to further check *automatically and periodically* for updated versions of said text files; receive automatically any updated versions of said text files in response to said program checking for said updated versions when said updated versions are available; then compile *automatically and periodically* using said combination of said updated versions.

But the above limitations have been addressed in claim 1.

As per claims 14-18, 20-24, these claims correspond to claims 2-6, 8-12 respectively, hence are rejected with the corresponding rejections as set forth therein, respectively.

As per claim 19, Bloch discloses text files particular to client system (e.g. pg. 4, para 0037; pg. 5, para 0047, 0050)

As per claim 25, Bloch discloses a computer-usable medium having computer-readable program code embodied therein for causing a computer system to perform a method comprising code for:

a) installing a program on said computer system (e.g. para 0050 pg. 5; para 0053-0055, pg. 6; AVM, Install File, starter page - Fig. 3; para 0057-0058, pg. 6 – Note: executing

installation code in AVM initializing stage including plugin/starter routines to setup memory or GUI handles – para 0062, pg. 7; Fig. 2-3 – reads on executing installed program on client), wherein said program comprises instructions for interacting with a server in accordance with:

installing on said computer system a plurality of text files wherein each of said text files defines a component of said application (*initializes ... downloads* - para 0055, pg. 6; para 0057-58, pg. 6; para 0061, pg. 7; step 70-72, Fig. 2, 5; para 0044, pg. 4; para 0063 pg. 7; para 0081-0083, pg. 9; *each function belongs to that application* – para 0085 pg. 9 – Note: download to incorporate content of XML file into application reads on installing plurality of text files defining the feel of the interface under development); and

compiling automatically (refer to claim 1) a combination of said updated versions of said text files to create said application, wherein said application is stored on said computer system in a compiled form so as to be executable independent from said program in a runtime environment independent from said program (refer to claim 1, and USC 112 Rejection); and

b) executing said application on said runtime environment independent from said program and independent from the program interacting with the server (refer to claim 1), such that said application remains executable on said runtime environment upon removal of said program from said client computer system (refer to claim 1, and USC 112 Rejection)

Bloch does not discloses executing the installed program to further check *automatically and periodically* for updated versions of said text files; receive automatically any updated versions of said text files in response to said program checking for said updated versions when said updated versions are available; then compile *automatically and periodically* using said combination of said updated versions.

But the above limitations have been addressed in claim 1.

Response to Arguments

11. Applicant's arguments filed 2/20/09 have been fully considered but they are not persuasive. Following are the Examiner's observation in regard thereto.

USC § 103 Rejection:

(A) Applicant has submitted that prima facie case has not been established as the Office failed to correctly address the intended language of 'executing … independently from said program …' particularly when Bloch virtual applications are construed 'on the fly' and being dependent upon network interactions (Appl. Rmrks pg. 9 middle). The current rejection has provided interpretation to the claim language in light of the many deficiencies, as set forth above in the rejection; and since the limitation as 'independently from' has not been defined with further requirements, the execution of the client application in Bloch is deemed meeting this execution as 'independent from' limitation, notably because, in Bloch, the intended client's development and validation of script data by way of the GUI interface with incorporation of updates into the modeling constructs is still ongoing even as the earlier instances of AVM code invocation to obtain server downloads have completed. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the reference.

(B) Applicant has submitted that the added limitation 'application remains executable ... upon removal ... of said program' distinguishes from Bloch's application that spawns from the AVM

memory (Appl. Rmrks pg. 10 top). The added limitation has necessitated another form of rejection; and would be moot in light of the changes in the Office Action.

(C) Applicant has submitted that Bloch's assembling of the application is temporary and tightly interacting with the AVM which when removed, would demise the temporary settings of such application, and that is different from executable program by a agent operating separate from the Application (Appl. Rmrks pg. 10, middle). The 'independent from' limitation has been addressed in section A; and the agent limitation is nowhere recited in the claim to impart any weight to the above rebut.

(D) Applicant has submitted that since the downloaded files are gathered to provide the appearance of the Virtual Application in Bloch, the assembled files (e.g. para 0047) cannot be dissociate from the realization of the Virtual Application; and paragraphs like 0086, 0100, 0102, 0103, 0105, 0109 are not evidences that code invoked are created by the AVM but that execute independent from the AVM (Appl. Rmrks pg. 11-12). The claim is not fulfilling a USC 112 requirement in many levels; and this description type deficiency is not permitting one to reasonably give weight to what appears to be Applicant's pleading that 'independent from' should be understood in terms that the main application while being assembled using files that have been retrieved from the external program, this very external program is disconnected from the runtime assembling of this application; and the reasons as to why only a certain way of interpreting the above claimed feature is identified in the currently effectuated USC 112 rejection. On top of the fact that 'independent from' limitation has been addressed in section A the limitation as to 'remains executable ... upon removal of said program' has also triggered another ground of rejection. In short, broad interpretation of 'independent from' has been

applied because of the many enablement issues detected when one attempts to impart proper merits to the elements recited in the claim. And until the enablement issues are resolved, the interpretation of the claim would be based on this outstanding deficient state of the claims (refer to the USC 112 Rejection), as set forth in the current Office Action. The arguments are therefore not persuasive.

In all, the claims stand rejected as set forth in the Office Action.

Conclusion

12. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on (571)272-3759.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.
Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR

system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Tuan A Vu/

Primary Examiner, Art Unit 2193

April 28, 2009